Software Engineering Institute

A Method for Assessing Technical Progress and Quality Throughout the System Life Cycle

Robert W. Ferguson
Summer C. Fowler
Rita C. Creel

November 2009

TECHNICAL NOTE
CMU/SEI-2009-TN-032

Software Engineering Process Management

Carnegie Mellon

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors thank Erin Harper for her expert technical editing and Dave Zubrow for his in-depth technical review.

Abstract

Project managers are responsible for numerous decisions affecting systems under their purview. To make these decisions and effectively manage projects, they need methods for quantifying technical progress and quality and identifying issues early in the project life cycle. Typically, information is obtained through reviews, which provide information on risks, problems, and work completed to date. However, these reviews are generally aimed at identifying problems within isolated components or artifacts. They often fail to expose problems related to the end-to-end system or the underlying, systemic issues that led to them.

The technical assessment method introduced in this paper addresses these shortfalls by providing a framework for evaluating a system from several perspectives, or views, for a comprehensive picture of progress and quality. It also describes a structured approach that customers and developers can use to measure progress at each review. A graphical indicator is introduced that clearly depicts technical progress and quality, enabling those involved with the project to make informed decisions to mitigate problems and reduce quality risks.

The results of applying this method to a software acquisition program are presented and an assessment of the effectiveness of the method is made. Recommendations for future use and study are also provided.

1 Introduction

As we consult with our acquisition customers and their development contractors, we often are asked how to evaluate progress and quality as early as possible in the development life-cycle. Practices such as Earned Value Management (EVM) enable analysis of cost and schedule data. Technical reviews provide information on individual system components. However, no widely used methods exist to quantify end-to-end progress and quality for the entire system.

While early life-cycle reviews provide some information on risks, problems, and work completed to date, they are typically aimed at identifying concerns and status related to isolated technical components rather than the integrated system. Additionally, they are not designed to identify systemic issues that may cause major problems downstream. Shortcomings in the way these technical reviews are conducted are listed below.

- Technical reviews are typically focused on individual deliverables and do not address the system as a whole, often failing to reveal inconsistencies or omissions between deliverables. As a result the technical review typically verifies adherence to component specifications, but not whether the system will operate as intended.
- Reviewers and developers often lack a systematic, quantitative approach to determine whether the system being developed meets operational needs and is displaying adequate technical progress and quality. Results of technical reviews are frequently presented as lists of comments, recommendations, and action items to be tracked to closure. Decision makers—customer and developer[1] alike—may have difficulty sensing the broader implications of the outcome or interpreting the significance of the list of actions, which may delay problem resolution.

A further challenge for software is a common but mistaken assumption that it is not possible to gain insight into software progress until coding and unit testing begin. Thus, software is frequently deemphasized or ignored at early technical reviews, enabling software-impacting defects in requirements, architecture, and design to propagate through the system with disastrous results.

The technical assessment method introduced in this paper addresses these shortfalls by

- providing a framework for evaluating the system from several perspectives, or *views*, for a comprehensive picture of progress and quality
- equipping the customer and developer with a structured approach for measuring progress at each review, beginning early in the life cycle, using a set of questions that address both satisfaction of technical requirements (i.e., meeting the specifications) and fitness for use
- presenting a graphical indicator that clearly depicts technical progress and quality, enabling the acquirer and developer to make informed decisions regarding risk and problem mitigation

Section 2 of this paper introduces the method, identifies its intended users, and suggests when it should be used. A trial application of the method to a software acquisition program is described in Section 3. Section 4 presents conclusions and next steps.

[1] In this paper, the term “customer” is used to refer to the acquirer, manager, or others responsible for the acceptability and use of the system in its intended mission. The term “developer” is used to refer to the development contractor or the development or engineering organization.

2 Assessing Technical Progress and Quality Using System Views

In the authors' experience, technical progress and quality assessments for software-intensive systems are often performed in a piecemeal fashion. The deliverable artifacts (e.g., documents, drawings, hardware, and software) are assigned to one or more subject matter experts (SMEs) for review and comment. Each SME prepares comments and recommendations according to the prescribed format and sends them to a review coordinator, who submits them to the development team for resolution.

While such an approach identifies problems and risks, it misses the mark in several key areas. First, cross-cutting concerns—those which span artifacts—are not explicitly targeted and are only identified if a reviewer happens to notice them. Such concerns are often the most complex and costly to resolve, so identifying and dealing with them early on should be a priority. Second, review results in the form of comments, recommendations, and resolutions do not answer the central questions, "Am I making adequate progress?" and "Are there significant technical issues that need resources now?" Finally, early life-cycle reviews do not identify software risks and problems that are observable in concepts of operations, requirements, and system architectures. Many assume these artifacts do not reveal anything about software, and the software design and (better yet) code must exist before software can be evaluated. This isn't true!

What Is the Focus and Contribution of this Method?

The assessment method we have developed leverages the concept of *system views* to provide an integrated picture of progress and quality. It focuses on technical reviews conducted throughout the life cycle. Early in the life cycle, these reviews center on operational concepts, requirements, and architecture products. As design, code, hardware, test products, and other artifacts become available they are brought into the review as well.

The method is effective because it

- adds rigor and consistency to the technical review process
- can be applied throughout the life cycle
- looks across products, synthesizing a variety of technical concerns into a single structure for communications between parties (e.g., customers and suppliers or systems developers and managers)
- addresses both the technical fitness of the product and systemic development process issues requiring management action (e.g., reallocation of resources)

In this way, the method identifies disconnects between stakeholders that other methods do not. These disconnects are often a source of serious progress and performance issues, making the ability to identify and resolve them as early as possible a significant benefit.

Is Additional Review Effort Required to Apply the Method?

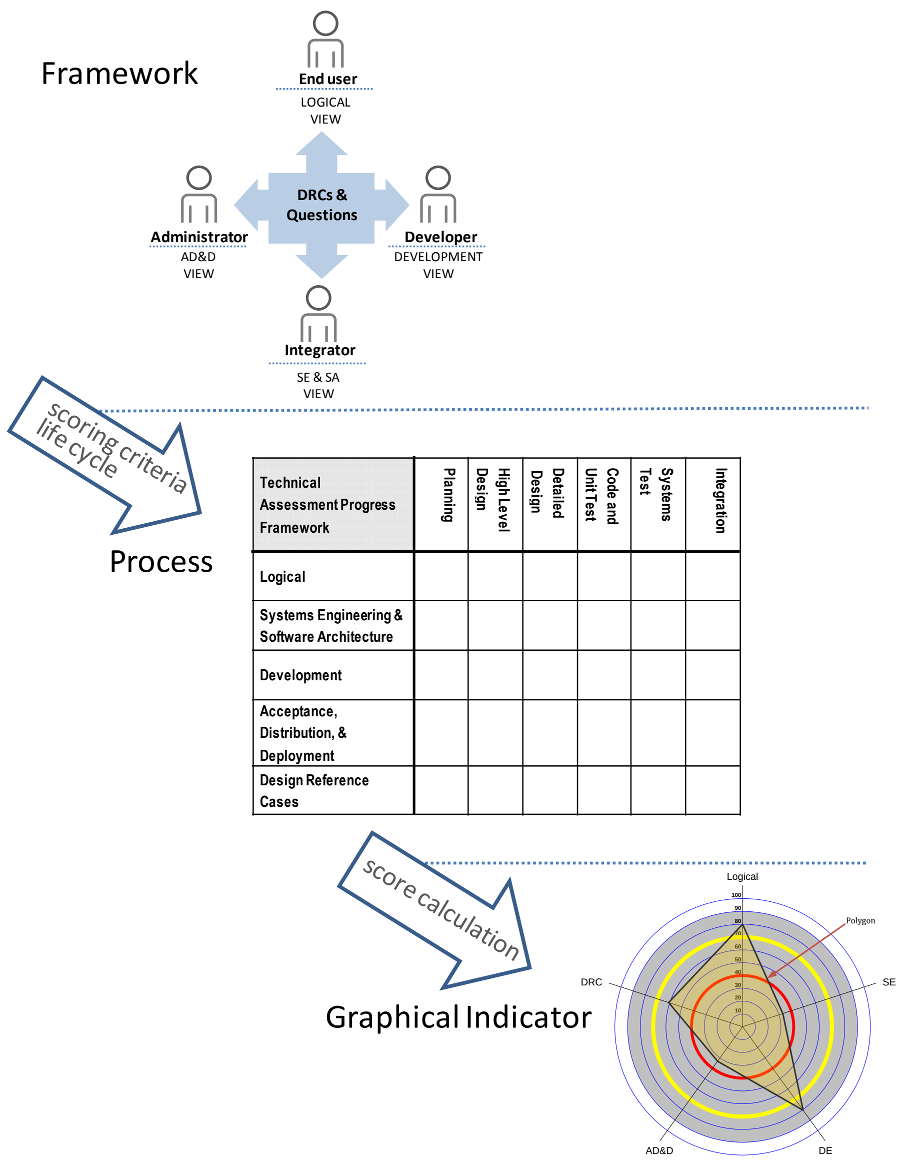
Based on our pilot application, the effort required to use this method appears comparable to that required for a less structured review. The difference is in the way resources are applied. Rather than looking at a single technical artifact at a time, participants are focused on the goals of a particular view (which may involve portions of several artifacts) and then on the convergence of the views into a comprehensive solution. As such, both reviewers and developers have significant responsibilities in preparing for and conducting the review.

What Are the Key Elements of the Method?

A robust technical assessment method requires the following three elements:

1. **framework for objective comparison.** Our method uses a framework of views, questions, design reference cases, and scoring criteria to provide structure and rigor to the review and to ensure that concerns that *cut across* artifacts are identified and addressed.
2. **process for assessment.** This process is designed to provide complete, quality results and identification of any risks or issues encountered in the assessment. Key process steps to prepare for, conduct, and complete the review are identified.
3. **mechanism for presentation, interpretation, and evaluation.** A graphical indicator of progress and quality is provided along with instructions for interpreting it and evaluating its integrity.

Figure 1 illustrates these elements and their interrelationships. The elements are described in detail in Sections 2.1, 2.2, and 2.3. Section 2.4 presents ways to determine if the method is having the desired effect.



Technical Assessment Progress Framework	Planning	High Level Design	Detailed Design	Code and Unit Test	Systems Test	Integration
Logical						
Systems Engineering & Software Architecture						
Development						
Acceptance, Distribution, & Deployment						
Design Reference Cases						

Figure 1: Elements of the Technical Assessment Method

2.1 Element 1: Assessment Framework

The technical assessment framework consists of the following components:

1. **views** that capture the interests of different stakeholders and are used to examine technical artifacts
2. **probing questions** for each view to verify coverage of design goals and existence of deliverable artifacts
3. **design reference cases (DRCs),** which are scenarios describing examples of performance drivers. The DRC is used to study how the design has implemented the scenario and how the developer has checked to see performance goal can be met.
4. **scoring criteria** that are used to assess and quantify how well the technical review deliverables meet the goals of each view

The following sections describe each component of the assessment framework.

Addressing the System from Multiple Views

Software architects will likely recognize the following table based on the work of Phillipe Kruchten [Kruchten 1995].

Table 1: Views Tailored for Technical Progress

View	**Purpose of View**
Logical	• focuses on the functional requirements by addressing the concerns of the system end user • assures alignment of the functional specifications with the intended use of the system
Systems Engineering (SE) and Software Architecture	• focuses on allocation of requirements and assigns design responsibility to subsystem/components. It addresses the concerns of the system integrator. • validates that system design meets performance objectives and other quality-attribute objectives (e.g., reliability, auditability, safety, security)
Development	• focuses on the system modules to address the concerns of the development team • coordinates team responsibility and component interfaces • addresses consistent use of development standards and processes • verifies and validates requirements allocation, staffing resource allocation, and cost estimation and planning
Acceptance, Distribution, and Deployment	• focuses on system readiness for deployment by addressing the concerns of the system administrator and support functions • verifies and validates manufacturing, installation, and support requirements

Table 1 describes four views of a system from the perspectives of different stakeholders. Participants will revisit these views during each technical review, which forces them to analyze the system *as a whole* beginning very early in the life cycle. The views also provide insight into non-technical issues that require management action, such as the discovery of a system module that is in need of more staff or staff with specific skill sets. This insight is obtained using pre-specified probing questions that participants answer by analyzing technical review deliverables.

Probing Questions

The second component of the assessment framework consists of probing questions used to assess system progress from the four views. Specifically, the questions are used to probe the technical review deliverables to provide a holistic indication of progress and identify areas that need focused attention.

Appendix A provides example questions that can be asked in relation to each view during technical reviews. Participants can tailor and use these questions, create their own, or use questions developed by other organizations. The idea of using questions to assess technical deliverables is not new. Many sample review checklists and questions have been developed by groups such as the Software Technology Support Center [STSC 2003] and the Software Engineering Institute

[Alberts 2009]. What is new about the approach described in this paper is that the questions are organized and analyzed in the context of the four views. This approach provides a deeper and broader level of insight into technical progress and quality.

Tailoring the questions for a system or project requires careful consideration of each view's stakeholders. A certain amount of expert judgment and experimentation are required to devise a set of questions that will cover the most critical aspects of each view at the correct level of detail.

Design Reference Cases (DRCs)

Design reference cases (DRCs) are sets of scenarios which are intended to focus on performance drivers (also called quality attributes) for a system [McLean 2006]. Unlike a typical use case, a DRC specifies minimum conditions for acceptable performance and is often used to describe how the system should respond when stressed. Typically, DRCs are created by systems engineers with input from users and stakeholders. The DRCs need to be specific and precise.

The format for a DRC is

context + stimulus = response + desired outcome

Some examples are shown in the bulleted statements below.

- [context] During heavy operation of the system, the system shall process [stimulus] ten thousand transactions within [response] one hour so that [outcome] no user experiences a delay of more than one minute.
- [context] During a storm [stimulus] a network node may be temporarily lost. [response] An operator must be able to recognize the lost node within 5 minutes, so that [outcome] maintenance personnel can be dispatched and service restored within 1 hour.
- [context] During normal system operation, the system shall [response] complete the response to [stimulus] an operator command [outcome] within three seconds.

Using scenarios to detect SE and software architecture problems has been explored before in the SEI's Architecture Tradeoff Analysis Method.[2] "Experience with Performing Architecture Tradeoff Analysis" is an experience paper on this method [Kazman 1999]. The method described here extends that work to many types of deliverable and multiple stages of the life cycle.

A project should develop several DRCs in order to describe a range of conditions, stimuli, and desired outcomes. Reviewers should select a small subset (i.e., three to four) DRCs for any given review in order to be able to complete the review in the allotted time. At a minimum the DRCs should cover a range of conditions as follows: 1) normal operations, 2) a stressing condition such as an out-of-bounds or unexpected event, and 3) maintainability, safety, security, and other relevant quality attributes related to system robustness. Figure 2 is an adaptation of Kruchten's 4+1 views that depicts the four views and the DRCs that span the views [Kruchten 1995].

2 The method is described on the SEI website at http://www.sei.cmu.edu/architecture/tools/atam/index.cfm.

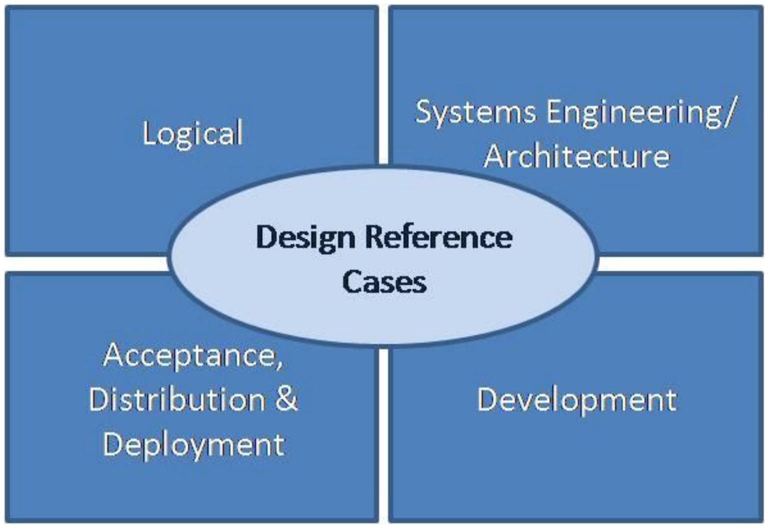

Logical
Systems Engineering/ Architecture
Design Reference Cases
Acceptance, Distribution & Deployment
Development

Figure 2: Design Reference Cases Span All Views

DRCs are used during the technical review meeting (described in Section 2.2) to walk participants through system operation. This method of conducting a technical review is not uncommon, but the use of the assessment framework is unique because it involves view-based questions rather than a simple walkthrough of each deliverable based on generic technical exit criteria. Answering and scoring the questions for the four views while the DRCs are demonstrated can indicate areas in which management needs to focus resources before system development progresses to the next phase of the life cycle. In other words, having both DRCs and the four views allows quantification of technical progress (using the scoring criteria) and identification of areas requiring a non-technical solution such as allocation of a subject-matter expert to work on a particular problem.

Note that as a project progresses through the life cycle and technical products become more detailed, it may not be possible to completely walk through each DRC. In such cases, the scope of the DRC evaluation should be narrowed to focus on the areas most significant to the review.

Scoring Criteria

Once DRCs have been defined and probing questions have been tailored for each view and for each technical review, it is important to establish a way to quantify the answers to the questions. *Scoring criteria* must be established to quantify how the answers compare to the *expected answers* for each view at the particular life-cycle stage under review. Scoring is not as simple as determining a "yes" or "no" answer for each question: objective criteria are important to the trustworthiness of the method.

Scoring Criteria for the Four Views

The scoring approach is based on the concepts of model-based verification where the DRC is the model and the probing questions are the claim [Gluch 1998]. This technique addresses both the formal need for objective observation and the pragmatic need for efficiency. Answers to probing questions are scored by determining whether the answer reveals

- a severe defect that causes a system to fail a specification or an inability to achieve desired outcomes (score = 3)

- an important issue (score = 1)
- a general question or concern that does not require action but is recorded for reference (score = 0)

The value "2" is not used in order to give more weight to failures (score = 3) but retain the ability to identify patterns that suggest process problems (score = 1).

Answers and scores are logged in a spreadsheet such as the one shown in Table 2. Score assignments for *severe defects* and *important issues* are illustrated for each architectural view. The decision as to whether something should be called a "severe defect" or an "important issue" depends on the context. Below are some examples. In your particular application of use, you may consider some of the "important issues" to be "severe defects" and vice versa.

Logical view

Severe defect (rating =3)

- A requirement or specification is vague or incorrect, or the interpretation of a requirement or specification in a component or design element is inconsistent across artifacts.
- The DRC indicates that the customer and the developer have a different understanding of the requirement.
- The proposed system design does not produce the desired outcome of the DRC.

Important issue (rating = 1)

- The design language is inconsistent, which may result in different interpretations by designers and developers.
- Inefficiencies in operation are revealed that affect business system throughput.
- The DRC does not appear to correspond to expected system use (i.e., it is a "bad" test case).
- Variances exist from any agreed upon naming convention or standard.

Systems engineering view

Severe defect (rating =3)

- The technology has not yet achieved the required technology readiness levels or key performance parameters corresponding to the current stage of development.
- A technology study is not complete or technical information is not complete (for example, unverified algorithms exist).
- A critical safety, compliance, or other assurance requirement does not appear in design work.
- System performance and capacity objectives are not quantified or the system does not achieve objectives.
- The external interface is missing or incorrect.
- A technical requirement has not been assigned to a specific design team or component.

Important issue (rating =1)

- Robustness scenarios for quality attribute(s) are missing or not satisfied, such as operations under stress, maintainability, dependability, fault tolerance, data integrity, or recovery after fault.

- The external interface is specified using different terminology in different documents (e.g., naming conventions differ or are not adhered to).
- An extra (i.e., unspecified) feature is identified that might be nice to have but increases development costs.
- Documentation standards or product data management standards are not followed, including naming and notation conventions (internal or external criteria).
- An expected process quality check is missing, such as an inspection, simulation, or other verification procedure.

Question (rating = 0)

- Are performance parameters needed for particular specifications?

Development engineering view

Severe defect (rating =3)

- Errors exist in the design or product (e.g., missing design element, design element that has no specification, or a function does not work or provides wrong answer).
- Internal interface errors exist.
- Serious test and verification failures occur, making the product unacceptable. The design does not fully implement the DRC (e.g., missing method, path, or data).
- failures occur at high severity levels, making the product unacceptable. The design does not fully implement the DRC (e.g., missing method, path, or data).

Important issue (rating =1)

- Failure to follow design process and standards (e.g., inconsistent application of design standards, configuration management problems, build problems, or skipping verification procedures such as simulation, inspection, and testing).
- Internal interface descriptions are not under configuration control or are not properly published and annotated.
- Complex design exists (coupling, inheritance, algorithmic complexity, or component functional size).
- Minor-to-moderate test and verification failures occur.
- Extra features are not specified by requirements, SE, or development standards.

Question (rating = 0)

- Is design responsibility assigned to the correct team?

Acceptance, distribution, and deployment view

Severe defect (rating =3)

- Planning for acceptance, distribution, and deployment have not been addressed.
- The product failed to achieve certification or is not ready for certification testing.
- A component cannot be acquired or delivered to the user in a timely fashion.
- The target environment is not identified or the product will not work in the target environment.
- Government furnished equipment/government furnished information (GFE/GFI) is not available for verification and validation when needed.

- The prototype from the subcontractor is not available.
- Implementation of the DRC in the proposed system design cannot be accomplished without major changes (this is most common in cases related to safety, security, auditability, or other quality attributes which impact the architecture, not just the design and implementation).

Important issue (rating =1)

- Installation or delivery requires extra, unplanned staffing and support.
- New training requirements are identified (e.g., additional user training or a system training function).
- Maintenance or other previously unidentified support function requires new staff, training, or a procedural change.
- Inconsistent naming and identification of support functions, installable items, or other logistics or training exists.

Questions or general concerns that are less critical than an important issue may be logged in the scoring sheet and given a rating of 0 for all views.

Scoring Criteria for the DRC View

The scoring procedure for the DRC view is described in this section. A DRC is said to fail if there is a major defect (i.e., a score of 3) recorded in any view. A DRC also fails if either the desired response cannot be verified or if the desired outcome cannot be validated. When the result is "fail" the error must be assigned to one of the views and recorded there. The maximum value of all findings for the DRC is identified in the scoring sheet. If that maximum is "3" the DRC fails; otherwise it is rated as "pass." For the DRC, we are interested in potential failures only. Minor errors are sufficiently recognized by the collection of data in the other views.

Because few DRCs can be fully validated before the system is fielded and operational, it is important that the scoring criteria reflect the current phase life-cycle phase. For example, at very early technical reviews the concept of operations and requirements may be the only deliverables available for review. The DRCs will still guide discussions, and scoring will be based on how well the available deliverables indicate that the system is making progress toward satisfying the DRCs. Answering the questions for each view as the DRC is demonstrated provides a way to identify systemic issues, for example, key performance parameters and design rules that can easily escape detection at early stages of the life cycle.

Accumulating the Scores

The following information about each observed defect, issue, and question or concern should be entered into a table:

- Review Date (date)
- Description of Defect (text)
- Defect ID (integer)
- DRC ID (integer) – note that the ID value may be blank when a defect is discovered with no corresponding DRC

- rating assigned to each architectural view (3 for severe defect, 1 for important issue, or 0 for question or concern)
  - Log: logical view
  - SE: systems engineering view
  - Dev: development engineering view
  - AD&D: acceptance, distribution, and deployment view
- DRC-val: rating assigned to DRC view, the maximum across all findings for the DRC

Scores should be recorded in a table similar to Table 2 below.

Table 2: Example of Defects Identified During a Technical Review Meeting

Review Date	Description of Defect	Defect ID	DRC ID	Log	SE	Dev	AD&D	DRC-val
01/01/2009	Timing diagram X has incorrect sequence	1	1	0	0	1	0	1
01/01/2009	Timing calculation (XYZ) is incorrect	2	1	0	3	0	0	3
01/01/2009	Specification p. 10 #3 is missing KPP	3		1	0	0	0	
01/01/2009	Defect #4	4	2	0	1	0	0	1
01/01/2009	Defect #5	5	1	1	0	0	0	1
01/01/2009	Defect #6	6	2	0	0	1	0	1
01/01/2009	Defect #7	7	2	0	1	0	0	1
01/01/2009	Defect #8	8	1	0	0	1	0	1

The first row of data in Table 2 shows an issue that is attributed to DRC #1 in the development view. While it is identified as an important issue ("1" in the Dev column), it does not cause the DRC to fail (indicated by the "1" in the DRC-val column).

The second row shows a defect that is attributed to DRC #2. This defect in a timing calculation is significant ("3" in the SE column) and does cause the DRC to fail (indicated by the "3" in the DRC-val column).

The third row shows a defect that was not discovered during the DRC walk-through (indicated by the blank in the DRC ID column) and is identified as an important issue for the logical view.

It is possible to rate a single defect in more than one view. This may imply that there are multiple actions needed to address the defect.

The additional rows of defects are used when generating the Excel pivot table and graphical indicator described in Section 2.3.

Assessment Framework Summary

This section described the four components of the assessment framework. *Views* cover the concerns of four types of stakeholders. *Probing questions* are used to investigate the technical review deliverables to determine if progress is being made toward the goals of the views. The *DRCs* are the scenarios used to identify concerns that span views and deliverables. As such, they focus on system operations and key performance drivers. Progress is quantified using the pre-determined *scoring criteria* that are adjusted for the phase of the life cycle in which the technical review meeting is held.

The sections below describe how technical reviews are executed using the assessment framework and how the scoring results can be depicted graphically for easy interpretation.

2.2 Element 2: Assessment Preparation and Execution Steps

The second element of the method consists of the steps needed to prepare for and execute the assessment of technical progress and quality. These steps include

1. planning the overall review strategy
2. preparing for each review
3. conducting each review
4. presenting findings

Each step is explained in more detail in the following sections.

2.2.1 Planning the Overall Review Strategy

This step involves planning the overall strategy for applying the method over the project life cycle. Planning the review strategy includes the following three steps:

1. Identify technical reviews.
   In this step, identify technical reviews to which the assessment method will be applied. The list of reviews should include both major technical reviews and, for the greatest benefit, selected informal reviews. Technical reviews might be listed in such documents as statements of work or project plans. Reviews should be selected to maximize potential for early identification of technical and non-technical issues that may adversely affect project success.
2. Identify and train developers and reviewers in the method.
   Developers and reviewers who will participate in the assessment must also be identified. Individuals should be selected based on expertise. Training in the method should be performed in advance of individual review preparation activities to allow sufficient time to construct DRCs and determine appropriate probing questions.

   All participants should be trained to use the assessment framework in Section 2.1. It is important to clarify that the purpose of each technical review is no longer merely to assess how well the deliverables meet the typical requirements of the review, but also to assess the progress and quality of the entire system's DRCs and to score questions in multiple views. For example, rather than assigning a reviewer to examine a single deliverable (such as a software requirements specification) to determine whether it meets standards and is adequate, the reviewer may be assigned a particular view and may need to look across multiple deliverables to answer the questions).
3. Construct design reference cases (DRCs).
   As stated above, DRCs are scenarios used to identify concerns that span views and deliverables and focus on system operations and key performance drivers. DRCs should be designed to stress the system. Stressing conditions can include normal operation under heavy loads, boundary and out-of-bounds conditions, or special performance criteria. Finally,

DRCs must represent the concerns of other "-ilities" such as maintainability, availability, security, and supportability. DRCs are developed in this step because they are used in each review throughout the project life cycle.

Once a preliminary list of technical reviews is available, participants can begin the process of preparing for individual reviews.

2.2.2 Preparing for Each Review

Preparation for the review involves data and information exchange between developers and reviewers. Typical roles are listed below.

- Developers provide a high-level overview of expected deliverables and relationships between them.
- Developers give reviewers access to all deliverables.
- Reviewers identify a subset of two to five DRCs that will, in turn, be provided to the development team.
- Reviewers tailor the framework for the project by studying the deliverables using the DRCs to develop probing questions and scoring criteria. Inputs from stakeholders representing each of the four views are essential in assuring that the questions cover the most critical concerns of that view.

Both the developer and reviewer should work from the same framework so all stakeholders understand what is to be evaluated during each technical review. Review scope is jointly established by reviewers and developers based on both the products for the review and the selection of DRCs. Reviewers must focus questions on work products available at the stage of the development life cycle under review. After a draft is complete, participants should check to ensure that both the DRCs and the questions address critical aspects of the system in a manner appropriate to the stage of the life cycle. They must also ensure that the DRCs are broad enough to cut across all four views.

Participants may need to adjust the DRCs, questions, and scoring criteria as system development progresses. These adjustments are not intended to change results but to enable greater accuracy in determining technical progress.

2.2.3 Conducting Each Review

The technical review meeting is conducted using the DRCs to demonstrate that the system is making progress toward satisfying the operational goals of the DRCs. Participants can confirm or dispute that technical progress is sufficient based on scores for the questions and evidence of support for the DRCs.

The steps listed below should be completed during the review.

1. Developers present their description of how the system treats each DRC.
2. Reviewers pose questions as needed.
3. Reviewers score the answers to the questions by identifying defects, issues, or questions/concerns not addressed.

4. The list of questions that are not satisfactorily addressed is used as a set of action items that identify areas where progress or quality is lacking.

This method of executing a technical review encourages active participation as participants walk through the deliverables to show how the system will satisfy the DRCs, which are now the focal point of the meeting. The DRCs provide participants with a way to show progress toward the technical goals of the system. For example, one of the DRCs might state that the system must handle 1000 transactions per minute during heavy traffic. Participants would walk through the deliverables (e.g., requirements specification or design documents) to look for evidence that the system is making progress toward this operational goal.

2.2.4 Presenting Findings

The output of each review is a set of scores for the questions and DRCs along with an agreed upon list of action items to address the questions that did not receive good scores. The scores of individual questions are integrated into a single picture called a Kiviat diagram that graphically depicts the technical progress of the system being developed to allow for quick interpretation. Any score of 3 or 1 should be supported by action items and findings. Scores of 0 (indicating questions) should generally also have associated action items. The reviewers then present recommendations for additional analysis or possible improvements to project management and the developers. The Kiviat chart is described in detail in Section 2.3.

2.3 Element 3: Graphical Indicators and Interpretation Guidelines

A Kiviat chart is used to communicate review results in a way that emphasizes progress toward meeting stakeholder needs. A Kiviat chart, such as the one shown in Figure 3, is used to represent several different measures for the same object, in our case, the technical review. Each radial axis represents a single measure and all axes are scaled to a common value (e.g., 1-100). The radial values are connected into a polygon. In a perfect review, the polygon would connect the maximum value on each axis forming the maximum area pentagon. As lower values are observed, less and less of the area is covered.

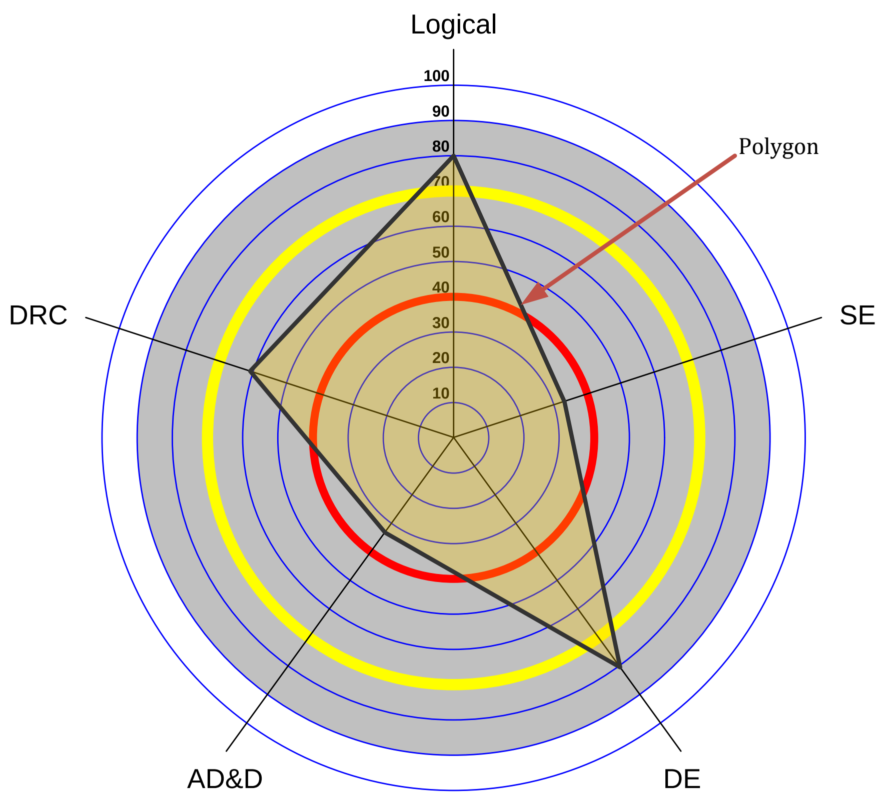


Figure 3: Example Graphical Indicator of Technical Progress

A Kiviat chart is generated using the results from the scoring method described in Section 2.1. The data from Table 2 were used to generate the Excel pivot table in Table 3, which has been populated with the values for each system view.

Table 3: Example Excel Pivot Table Generated Using Table 2 Data

DRC ID	**Count**	**Sum of Log**	**Sum of SE**	**Sum of DE**	**Sum of AD&D**	**Max of DRC-Fail**
1	4	1	3	2	0	1
2	3	0	2	1	0	0
	1	1	0	0	0	0
Grand Total	**8**	**2**	**5**	**3**	**0**	**1**

In Table 3, each row represents either a DRC, identified by the DRC ID, or the collected defects that are not associated with a DRC (in this case, the DRC ID is blank). The columns are the sum of the values for each column (LOG, SE, Dev, and AD&D) in Table 2. The “grand total” values are the sums of the cells above.

The Kiviat chart contains 5 axes, one for each view and one for the DRCs. The value for each axis is normalized from 0-1.00 with 1.00 representing a perfect score. The grand total (sum of scores) in each column is used to calculate the score for the corresponding axis using the following formula:

Axis-score = 1 – (sum of scores in column) / (3 * number of rows in table)

Since the findings table has 8 rows and the "Sum of Log" column has a value of 2, the calculation for the logical axis is 1 – (2 / (3*8)) = 0.92

The DRC axis value is normalized to the same 100% scale using DRC = 1 **-** ((#failed DRC)/#DRC). Kiviat axis values are shown for all four views and the DRCs in Table 4.

Table 4: Kiviat Axis Values for the Four Views and the DRCs

Logical	SE	Dev	AD&D	DRC
0.92	0.79	0.88	1.00	0.50

In this manner all results can be plotted on the same scale and shown as a Kiviat chart, such as the one in Figure 4. Both the technical teams and project management can use this chart as a way of identifying and quickly communicating where problems exist.

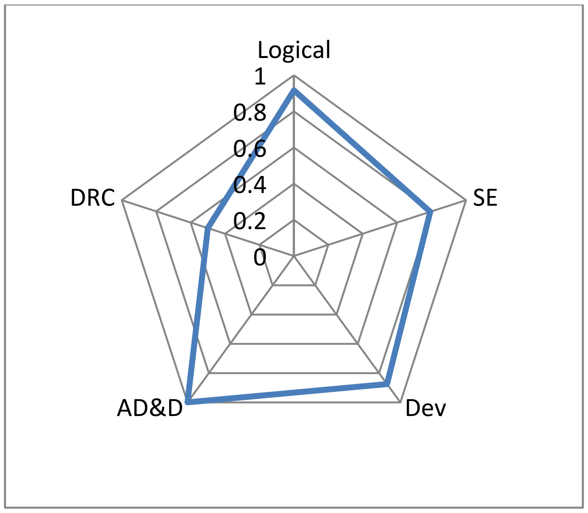


Figure 4: Kiviat Chart for Views and DRCs

2.3.1 Interpreting the Diagram

When interpreting Kiviat charts, one expects the most failures and the worst scores to be associated with current work, since it is the least mature. For example, during coding and testing, more problems should appear in the development engineering view (Dev).

If, however, a large number of defects is associated with past work (e.g., in the SE view), it is possible that some facet of high-level design is incomplete or was poorly executed, or that significant requirements changes have been made. These sorts of concerns, representing earlier life-cycle artifacts, will likely result in significant rework and will be more costly to correct.

DRC Scores

Figure 4 shows that half the DRCs failed. Efforts to investigate and resolve the cause, or causes, will likely span deliverables and views. The benefit is that the failures have been discovered as early as possible in the life cycle, reducing cost and time to remedy.

Problems in the Logical (Log) View

Problems attributed to the logical view suggest problems with system definition and requirements processes. Such problems may begin with the requirements and concept of operations. Problems in the logical view may also be a result of changes in personnel or mission from the customer's side. From the developer's side, problems may result from a misunderstanding of the target problem domain.

Action responses must involve the developer and customer jointly. If possible, the customer should provide access to user personnel as well.

Problems in the Systems Engineering (SE) View

Undetected SE problems eventually show up as failures to achieve system performance and quality goals. Systemic problems associated with SE work relate to the technical solution, the project structure (i.e., the relationship between the system component structure, team structure, and project schedule), and systems engineering processes.

The SE view also provides an early means to detect misalignment between the system component structure and the structure of the development organization. When the two are not aligned, it causes decision delays and conflicts in component responsibility. Fixing or preventing these problems is a relatively new subject in complex development work. One tool for further analyzing these problems is the design structure matrix [Smith 1997].[3]

Problems in the Development (Dev) View

Systemic problems in development are often due to one of the following: incomplete implementation of a DRC, inconsistency in interpretation of design, or conflict in interface definitions.

Several problem types that can contribute to development engineering problems include the following:

- missing or skipped standards, inspections, and other verification and validation procedures. This includes static analyses that evaluate complexity and dynamic analyses that can identify problems such as memory leaks and bottlenecks. Orthogonal defect classification, developed by IBM, is a useful method of identifying and resolving these problems.[4]
- process problems in configuration management and component integration. Masashki Iisuka discusses this topic in the Fujitsu journal *FSTJ*.[5]
- team performance problems and integrated product team performance problems that are management problems rather than technical problems

Problems in the Acceptance, Distribution and Deployment (AD&D) View

The AD&D view allows reviewers to verify reference cases for distribution and deployment during early stages of development. It also invites reviewers to validate the suitability of proposed support functions, facilities, and operations as these functions often require separate classes of user. If the support mechanisms must change in order to deploy the new system, the changes must be planned and training developed prior to deployment.

[3] See http://www.dsmweb.org for more information.

[4] See http://www.research.ibm.com/softeng/ODC/ODC.HTM for more information.

[5] This paper is available at http://www.fujitsu.com/downloads/MAG/vol42-3/paper09.pdf.

Systemic problems in either the logical or AD&D view are representative of problems that do not belong solely to the developer. Developer and customer typically must work together to devise strategies to address these problems.

2.4 Evaluating Effectiveness of the Assessment

The following questions may be used to determine if the assessment is having the desired effect.

- Are reviewers taking a more active role in technical reviews?
- Have reviewers prepared for the review appropriately? Have they identified DRCs? Have they identified probing questions?
- Are defects/issues being identified earlier in the project life cycle? Has phase containment of defects improved?
- Were the findings used as described in Section 2.3?
- Was the review meeting objective in its execution?
- Were results accepted by all parties? If not, why not?

To provide more insight into how the assessment is applied and evaluated, Section 3 describes the initial application of this method on a large-scale software-intensive system.

3 Trial Application of the Method

A trial application of the method described in this paper took place during technical reviews prior to a preliminary design review (PDR) of a large-scale project with over three million lines of software code. After an initial briefing to describe the proposed assessment method, the review team from the acquirer's program office agreed to participate in a trial use of the method. The evaluation was based on a single subsystem considered to be one of the larger and more complicated in the project. This subsystem had to be redesigned to accommodate major additional functionality. The trial was expected to provide a reasonable opportunity to judge the cost and effectiveness of the method and to provide additional information in preparation for wider adoption of the assessment method.

3.1 Preparing for and Executing the Assessment

Because this was a trial, all of the recommended steps were not fully executed. Any deviations will be noted in the sections below.

3.1.1 Planning the Overall Review Strategy

Identifying Technical Reviews

Since technical reviews were already scheduled when the trial began, this part of the method was not executed.

Identifying and Training Developers and Reviewers in the Method

The acquiring project director determined that the subsystem software lead and her technical team of systems and software engineers should participate. The SEI supported this effort by training project management office (PMO) participants in the assessment method and actively participating in the technical reviews. All PMO participants were briefed on the three elements of the assessment method (described in Section 2), and the SEI support team was given all the engineering deliverables related to the subsystem being assessed, including materials from previous technical review meetings. This overview training was accomplished in one day for about ten people. Two additional training days were provided for four engineers. These additional days were committed to identifying the critical questions to be addressed during the course of the review procedure and reviewing the available design reference cases.

The lead engineer determined that the prime contractor's software development team would not participate in this trial in order to minimize any potential impact on contractor resources. See Section 3.2 for a discussion of the impacts of this decision.

Constructing DRCs

The subsystem being assessed was already in operational use and was undergoing an upgrade to include new functionality and provide additional throughput. The purpose of the technical review was to assess whether the new design was sufficient to pass PDR and to identify any gaps in the design. The program had created a number of DRCs as a normal part of program practice prior to involving the SEI. An existing operational scenario focusing on timing was selected because performance is a critical requirement of this system. That scenario served as the DRC for planning the assessment.

3.1.2 Preparing for the Technical Review

The next task was to define the probing questions for each view that would exercise the DRC. The team populated a spreadsheet with questions appropriate for the review. These questions were similar to the example questions provided in Appendix A for each view in the high-level design. Questions included the following:

- Is the mapping of requirements to the architecture complete and consistent? (logical view)
- Do timing views show thresholds for acceptable performance? (systems engineering view)
- Do we have size estimates for each computer software component (CSC)? What is the estimated versus actual number of CSCs? Has each CSC been assigned to a development team? Which components show the highest levels of coupling in the design? (development view)
- Have we defined the number of installable software units per site? (deployment view)

Several questions for each view were defined and documented to be distributed to the team for use when reviewing project deliverables.

Reviewers were selected based on their individual areas of expertise. However, rather than asking reviewers to assess a particular deliverable such as a software design description, they were asked to answer the questions of a particular view and to assess if there were any areas of the DRC that were not satisfied. Answering the questions in most cases required looking at multiple deliverables and talking to the prime contractor development team.

No additional time accommodation was made for the trial of the new review method. Allotted calendar time for preparation was fixed based on experience with prior reviews.

3.1.3 Conducting the Technical Review

Developers and reviewers conducted several technical meetings prior to the actual review. These additional meetings provided opportunities to adjust the questions and helped them interpret and analyze the design documents. In the course of documenting the questions, it became obvious to the reviewers that a second scenario was needed to rigorously exercise the new design elements. This second DRC also addressed concerns about system performance but via distinctly different control and timing parameters.

Reviewers then independently used the two DRCs to trace system flow, timing, and class diagrams. The DRCs exposed a number of inconsistencies and some potential missing items. Inconsistencies, errors, and missing items were recorded as a set of questions to be used during the formal review.

The customer’s chief engineer was concerned that the trial of the method would extend the schedule and affect the timing of the planned PDR. This led to another deviation from the recommended steps in Section 2.2. The reviewers had not attempted to influence the agenda and construct of the review meetings to match the assessment method, so the contractor development team presented the material component by component rather than using the DRCs. The style of the component-level review was to show how the new features and specifications required new modules or changes to legacy designs.

During the formal review meeting, the reviewers used their notes to introduce questions and findings as the corresponding diagram was discussed. Every time the answer indicated an inconsistency or problem, a corresponding action item was recorded. Following the technical review meeting, reviewers used the notes and action items to identify and categorize the defects as shown in Table 5.

Table 5: Defect Data from PDR

Review Date	Description	Item	DRC-id	Log	SE	Dev	AD&D	DRC-val
4/13/2009	Specification p15.#8 appears to be missing KPP	1		1	0	0	0	1
4/13/2009	Class diagram QMMKLS contains objects not consistent with naming convention	2	1	0	1	0	0	1
4/13/2009	Class diagram QVCLX is missing method for CMDTRV	3	2	0	1	0	0	1
4/13/2009	Cannot verify timing throughput for CMDTUN1	4	2	0	1	0	0	1
4/13/2009	Class in diagram CMDILS misnamed	5	1	0	0	1	0	1
4/13/2009	Class in diagram CMDILS arrow wrong direction	6	1	0	1	0	0	1
4/13/2009	Timing diagram QEP21 has wrong sequence	7	1	0	0	1	0	1
4/13/2009	Timing EQA calculation wrong	8	2	0	3	0	0	3
4/19/2009	440 purging not a pertinent request	9	1	0	0	1	0	1
4/19/2009	Timeout of 420 req needs to wait to send transmission	10	1	0	0	1	0	1
4/19/2009	2 new data stores for channels missing	11	2	0	1	0	0	1
4/19/2009	Change needed for SWICD table 4 diagram (SE)	12	1	0	1	1	0	1
4/19/2009	Change SW SWICD interface to either sys A or sys B	13	2	0	1	1	0	1
4/19/2009	Active channel may be locked by intermittent problem. Stop sending requests.	14	2	0	3	0	0	3
4/19/2009	Can a timing request be interrupted?	15	2	0	1	0	0	1
4/22/2009	Wrong name for ML log file	16	1	0	0	1	0	1

Table 5 shows that two DRCs were used for the reviews. A total of 16 defects were identified, with two classified as being severe (defects #8 and #14, both in the systems engineering view and related to DRC #2). As a result DRC #2 received a failing value (i.e., 3) for the defects. There are several important issues noted by the defects with a score of 1, but these did not cause the DRCs to fail (as noted by the 1s in the DRC-val column).

Based on this defect data, several action items were recorded and assigned. It is important to note that the process of assigning the action items is facilitated using the views.

3.1.4 Presenting Findings

The pivot table and Kiviat axis values shown in Figure 5 and Table 7 were generated using the defect data in Table 5.

Table 6: Pivot Table Created from Table 5 PDR Defect Data

DRC ID	Count	Sum of Log	Sum of SE	Sum of DE	Sum of ADD	Max of DRC-Fail
1	8	0	3	6	0	0
2	7	0	11	1	0	1
	1	1	0	0	0	0
Grand Total	**16**	**1**	**14**	**7**	**0**	**1**

Table 7: Kiviat Axis Values for Review Defect Data

Logical	SE	Dev	AD&D	DRC
0.98	0.71	0.85	1.00	0.50

Using this data, we generated the Kiviat chart shown in Figure 5.

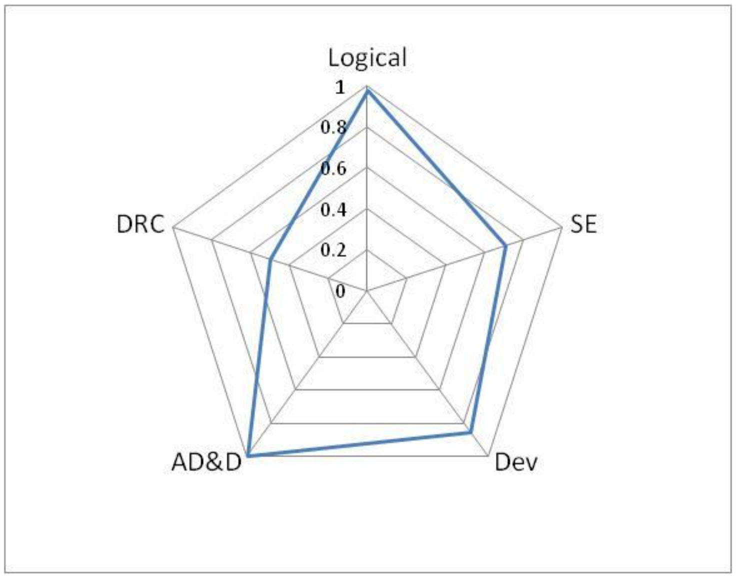


Figure 5: Kiviat Chart for Review Results

Figure 5 shows that half of the DRCs failed, indicating a technical problem with the system. In this example, the technical problems are the incorrect timing calculation (defect #8) and the “channel busy” problem (defect #14). Management can use this chart to quickly note that action is necessary to correct the defects before proceeding to steps beyond PDR. They can also use this chart as a starting point to identify where the defect originated by investigating processes in systems engineering and development. In general, the outcome of this particular review is satisfactory. The problems (SE and Dev) are typical of problems that should be caught at this stage of the life cycle, items prior to the final PDR. If problems had been associated with the Log view instead of the SE view, there would have more cause for concern, as these reflect problems with earlier life-cycle artifacts and processes.

3.2 Effectiveness of the Assessment Method

The three reviewers responsible for leading the trial of the method spent another day assessing the method itself. The main concerns were the effort required and the effectiveness of the method.

The reviewers concluded that calendar time required for the review was unaffected, but that the review team's focus had switched from the objectives of each individual deliverable to the consistency and completeness of the overall design. All three reviewers felt the method provided for deeper and more thorough analysis of the development products because of the tailored questions that each reviewer used. They noted the following improvements provided by the method:

- Reviewers were more active participants during the technical meetings.
- Reviewers asked more detailed questions about the product under review, making coverage of deliverables both broader and deeper during the meetings.
- Reviewers easily identified inconsistencies between deliverables.
- Reviewers identified some items missing from products under review.
- Use of scenarios accomplished the intended objective of assessing the consistency and completeness of the design.
- Areas of suspected complexity and risk were more readily identified and probed.
- Reviewers had high confidence in the results as presented to management.
- The team could easily communicate results to senior management.

Considering whether to use the new method on future reviews, the participants recognized the following opportunities for improvement:

- participation by development team

  The reviewers noted that the method would be more effective if both the acquiring team and the development team were trained to use the method. In this trial only the acquirers used the method, so the technical meeting itself was not conducted using the scenarios. While this did not prohibit examination of the deliverables using the scenarios and questions, it was recognized that many items would have been addressed by the development team prior to the technical meeting if they were also using the assessment method.
- formalize procedure

  It was noted that the procedure for tailoring the framework to a project should be formalized to improve training and execution of the method. Having guidance on how to select or develop scenarios with the appropriate level of detail would also be helpful. For the trial, only a single scenario was selected initially. The second was introduced later in the review cycle to account for new functionality, but it also addressed normal operations. Reviewers recognized that it would have been beneficial to select additional DRCs that accounted for system outages and maintenance activities.
- technical meeting execution

In addition to having the development team use the method, the review team agreed that the technical review meetings should be conducted using the DRCs to present the design. Using DRCs covering normal operations, a stressing condition, and potential enhancements, the review team could examine how well the design satisfies each DRC and to identify areas where both technical and non-technical gaps may exist.

- validating scoring procedure

  The team did not use the recommended scoring procedure during the review. The defect list, pivot table, and Kiviat chart were generated after the meeting. The team did review the scoring procedure, and they agreed that both the scoring and graphical indicator would be beneficial in future technical reviews. They also noted that validating the scoring process would be required for continued application of the method.

Summary

The reviewers stated that the method was effective because it identified issues earlier than they would have been identified using current methods—especially in the development view, where they most likely would not have been discovered until a later review (e.g., a CDR versus a PDR). They did note several areas that could be improved and determined that these improvements, especially training the development team and executing the technical meetings using the method, would require additional effort. However, all reviewers agreed that once the team was trained and using the method, the benefits would exceed the costs of any additional effort.

4 Conclusions and Next Steps

The assessment method introduced in this paper provides a way to quantify technical progress and quality and to identify issues as early as possible in the project life cycle. The method consists of three main elements: a framework consisting of four views, probing questions, design reference cases, and scoring criteria; assessment preparation and execution steps; and a graphical indicator depicting technical progress and indicating areas that need either technical or management action. We conducted a trial application of the method in a technical review of a component system leading up to the preliminary design review (PDR) of a large-scale software project. Using this method several specific design issues were discovered that would have otherwise not been identified until later in the project life cycle. The results of the review enabled the team to clearly and defensibly communicate satisfactory technical progress to upper management. The reviewers provided several suggested improvements to the method but unanimously agreed that the method was effective at quantifying and communicating project technical progress.

As noted in Section 2.1, a more formal process for developing probing questions and design reference cases needs to be documented. The process will aid teams as they tailor the framework for their respective projects and develop questions that capture the appropriate level of detail for the view at each phase of the project life cycle. Additionally, several suggestions from the participants in the trial will be incorporated in subsequent applications of the method. Extensions of this work are currently in progress, including an approach to using the framework and views as a communication tool [Fowler 2009] and a paper by Robert Ferguson titled *A Method for Leading Indicators for Program Management* to be published in 2010.

The scoring method and the Kiviat display need validation. The normalization procedure may hide, rather than reveal, certain concerns if there are a large number of findings. The effects of summing up results of several technical review meetings for a CDR-type review has not been tested. The authors also plan to make a more positive connection to the model-based validation technique.

Finally, the current method does not facilitate a comparison of Kiviat charts over a period of time. Any particular review has the potential to have a perfect score and does not predict a similar high score for subsequent reviews. In the future, some means to visually demonstrate progress and keep the same detailed technical insight should be developed.

Appendix A: Developing the Questions

This appendix provides sample questions that can be used with this method. The reader may tailor these, develop new questions, or use questions from other sources, including *A Framework for Categorizing Key Drivers of Risk* and the STSC [Alberts 2009 and STSC 2003].

Questions for Collected Deliverables

- What input documents, requirements, etc., were used to develop these deliverables?
- What features and functions are addressed by the current set of deliverables?
- Which teams developed the deliverables?
- Which teams must use the deliverables in their work?
- What performance requirements must be satisfied by the deliverables?
- What scenarios will exercise the intent and boundaries of the functions covered?
- What scenarios covering potential system breakdown are appropriate to the deliverables.

Logical View Questions

Verify the features and functions described.

- Do the deliverables implement the required features and functions?
- Are any gaps in functionality revealed in the scenario evaluation?
- Are connections to external systems properly specified for use by other designers and engineers?

Validate the use of the features and functions described.

- Is the language of the features and functions consistent with the domain language of the users?
- Are there any functional inconsistencies revealed in scenario evaluation?
- What characteristics of the system make use more difficult? Learning more difficult?
- Are operational requirements addressed?
- Have any users been ignored? Has any context of use been ignored (operation under fire, while mobile, etc.)?

Systems Engineering View Questions

Verify coverage of quality attribute goals and existence of measures for quality attributes.
Verify assignment of system functions and features to subsystems and components.

- What are the quality attribute concerns (performance, safety, availability, auditability, etc.) of the deliverables?

- What scenarios have been used to address those concerns? (Two approaches to developing these scenarios include the SEI Architecture Tradeoff Analysis Method[6] and the Quality Attribute Workshop[7]).
- What studies (trade studies, prototypes, simulations, etc.) have been used to address performance criteria?
- Have functional responsibilities been assigned to system components and software architecture patterns (e.g., layers)? What systems engineering decisions are dependent on each other (e.g., power – weight)? Does design indicate that these decisions are settled or that they are still in flux?
- Are ICDs defined to the appropriate level for the current deliverables?

Validate the capability of the system to meet quality attribute goals.

- Can scenarios be mapped to design components to study how design addresses quality attribute concerns?
- Have studies demonstrated ability to achieve performance goals? Include hardware prototypes and simulations and other examples of artifacts used to demonstrate technology readiness.[8]
- Have developers been appropriately informed of assurance criteria (safety, security, auditability, etc.)?
- What analysis work is being repeated because detailed design identified additional questions?
- Is there evidence of conflicting design efforts among development teams (i.e., two groups attacking the same problem)?
- Do subsystems or components have consistent names and configuration management structures?

Can systems engineers use the results of verification activities to validate achievement of quality goals? Identify system and project risk factors.

- What factors affect system and project complexity (e.g., number of components, number of interfacing systems, overall size of system, or number of components per feature)?
- Does system complexity analysis reveal high complexity at specific nodes of the system? Identify measures of coupling by components and dynamic dependency questions.
- Do development teams have the capability to use the planned technology? Does a change to the technology affect the development process?
- During testing, do systems engineers look for evidence of component complexity and high failure rates?
- Are systems engineers involved during deployment to gain an understanding of problems that arise during deployment and acceptance?

[6] See http://www.sei.cmu.edu/library/abstracts/reports/00tr004.cfm for more information.

[7] See http://www.sei.cmu.edu/library/abstracts/reports/03tr016.cfm for more information.

[8] Stevens Institute of Technology also provides technology readiness questions on their website for the Systems Development & Maturity Laboratory (SD&ML). See http://www.systemreadinesslevel.com for more information.

Development View Questions

Verify design implements required features.

- Are design components and variables identified (named) consistently?
- Can scenarios be traced through components, including calling structure and messages? Are missing and incomplete component documents identified?
- Do ICDs exist for interfaces? Do they cover requirements for interfacing components?

Validate deliverable for testers, support personnel, and users.

- Can testers build test cases from deliverables and scenarios?
- Does design language appear to interpret scenarios with consistent meaning?
- Does design implementation satisfy the users?

Are measures identified for the following?

- system size growth (component counts, software size)
- component complexity (functional responsibility, verification of algorithms, complex component paths)
- unit testing results
- failure to follow internal development standards (comments, naming, entry, exit)

Deployment View Questions

Verify that the design covers implementation and support concerns as needed for current deliverables.

- Are new support requirements identified? Are prototypes ordered (if necessary)?
- Are quality attributes and assurance scenarios defined and verified?
- Do system test cases cover requirements? Are acceptance criteria defined?

Validate designed system for users and support teams.

- Are users and support teams identified and represented? Are support technical roles and training identified?
- Does design language interpret deployment and support scenarios with consistent meaning?
- Does design meet criteria for support, maintenance, and deployment?
- Are system test cases and acceptance criteria agreeable to users and support teams?
- Are measures identified for the following?
  - number of installations, geographical growth, etc.
  - technology readiness levels, technology transition for users, and deployment
  - integration and system test results and defect queue
  - time required for installation and startup

References

[Alberts 2009]

Alberts, C. & Dorofee, A. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tr007.cfm

[Fowler 2009]

Fowler, S. & LoPresti, A. "Do I Have to Learn Klingon? Closing the Communication Gap with Your Technical Team." Project Management Institute (PMI) Global Congress Publication, 2009.

[Gluch 1998]

Gluch, D. & Weinstock, C. *Model-Based Verification: A Technology for Dependable System Upgrade*, (CMU/SEI-98-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 1998. http://www.sei.cmu.edu/library/abstracts/reports/98tr009.cfm

[Kazman 1999]

Kazman, R., Barbacci, M., Klein, M., Carrière, S. J., & Woods, S. G. "Experience with Performing Architecture Tradeoff Analysis." *Proceedings of the 21st International Conference on Software Engineering,* Los Angeles, California, United States, May 16 - 22, 1999. ICSE '99. ACM, New York, NY, 54-63.

[Kruchten 1995]

Kruchten, P. "Architectural Blueprints – The "4+1" View Model of Software Architecture." *IEEE Software 12,* 6 (1995, November): 42-50.

[McLean 2006]

McLean, Ian S. & Casali, Mark M., eds. "Ground-Based and Airborne Instrumentation for Astronomy II." *Proceedings of SPIE the International Society for Optical Engineering 6269*, Orlando, Florida May 25-29, 2006. SPIE: Bellingham, WA.

[Smith 1997]

Smith, R. & Eppinger, S. "Identifying Controlling Features of Engineering Design Iteration," *Management Science 43,* 3 (March 1997): 276-293.

[STSC 2003]

Software Technology Support Center (STSC). *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems*, 2003.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE November 2009		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE A Method for Assessing Technical Progress and Quality Throughout the System Life Cycle			5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Robert W. Ferguson, Summer C. Fowler, Rita C. Creel			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2009-TN-032
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK, 5 Eglin Street, Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE
13. ABSTRACT (MAXIMUM 200 WORDS) Project managers are responsible for numerous decisions affecting systems under their purview. To make these decisions and effectively manage projects, they need methods for quantifying technical progress and quality and identifying issues early in the project life cycle. Typically, information is obtained through reviews, which provide information on risks, problems, and work completed to date. However, these reviews are generally aimed at identifying problems within isolated components or artifacts. They often fail to expose problems related to the end-to-end system or the underlying, systemic issues that led to them. The technical assessment method introduced in this paper addresses these shortfalls by providing a framework for evaluating a system from several perspectives, or views, for a comprehensive picture of progress and quality. It also describes a structured approach that customers and developers can use to measure progress at each review. A graphical indicator is introduced that clearly depicts technical progress and quality, enabling those involved with the project to make informed decisions to mitigate problems and reduce quality risks. The results of applying this method to a software acquisition program are presented and an assessment of the effectiveness of the method is made. Recommendations for future use and study are also provided.			
14. SUBJECT TERMS technical progress, assessment method, system evaluation, views, Kiviat chart			15. NUMBER OF PAGES 44
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102